Relating and Visualising CSP, VCR and Structural Traces

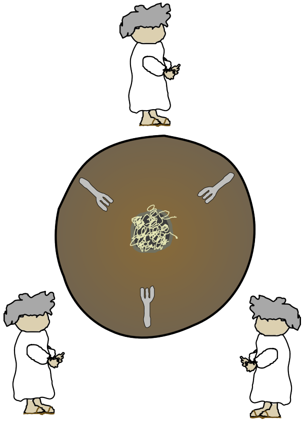
Neil Brown[1]   Marc Smith[2]

[1]Computing Laboratory, University of Kent, UK

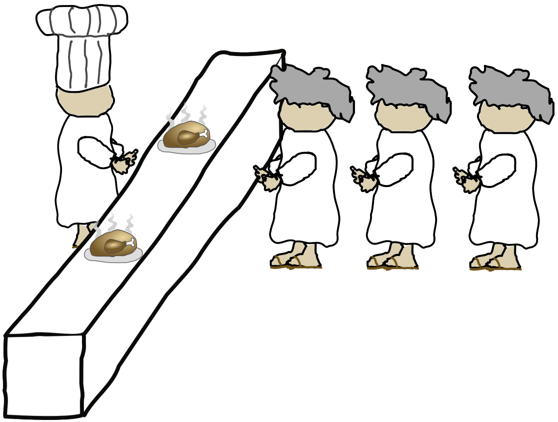
[2]Department of Computer Science, Vassar College, NY, USA

2 November 2009

Dining Philosophers



Starving ~~Dining~~ Philosophers



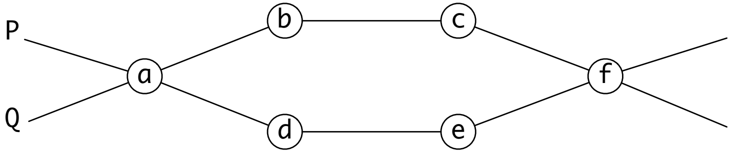
Visualization: Strings and Beads

Consider the CSP system:

$$P \underset{\{a,f\}}{||} Q$$

where

$$P = a \rightarrow b \rightarrow c \rightarrow f \rightarrow \text{SKIP}$$
$$Q = a \rightarrow d \rightarrow e \rightarrow f \rightarrow \text{SKIP}$$



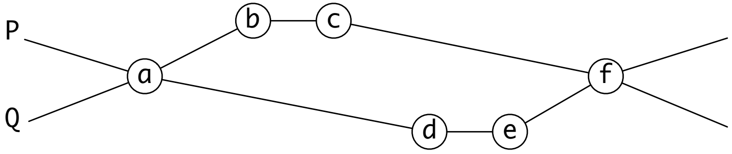
Visualization: Strings and Beads

Consider the CSP system:

$$P \underset{\{a,f\}}{||} Q$$

where

$$P = a \rightarrow b \rightarrow c \rightarrow f \rightarrow \text{SKIP}$$
$$Q = a \rightarrow d \rightarrow e \rightarrow f \rightarrow \text{SKIP}$$



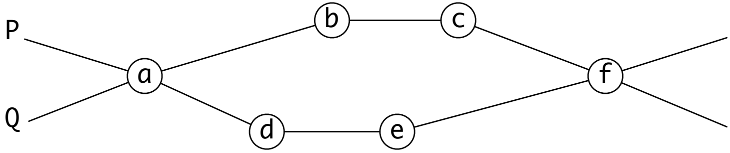
Visualization: Strings and Beads

Consider the CSP system:

$$P \underset{\{a,f\}}{||} Q$$

where

$$P = a \rightarrow b \rightarrow c \rightarrow f \rightarrow \text{SKIP}$$
$$Q = a \rightarrow d \rightarrow e \rightarrow f \rightarrow \text{SKIP}$$



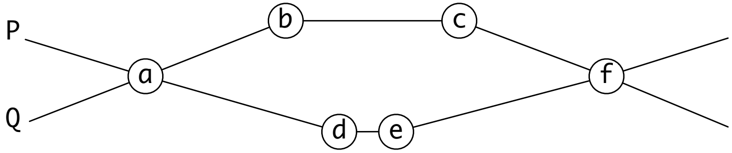
Visualization: Strings and Beads

Consider the CSP system:

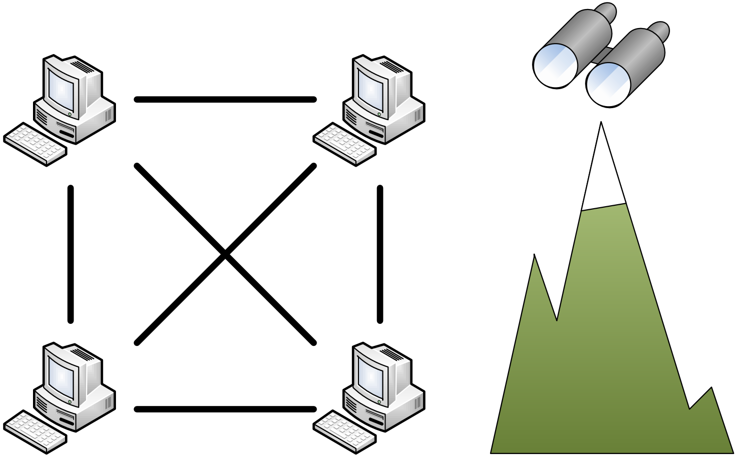
$$P \underset{\{a,f\}}{||} Q$$

where

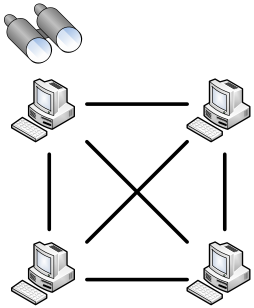
$$P = a \rightarrow b \rightarrow c \rightarrow f \rightarrow \text{SKIP}$$
$$Q = a \rightarrow d \rightarrow e \rightarrow f \rightarrow \text{SKIP}$$



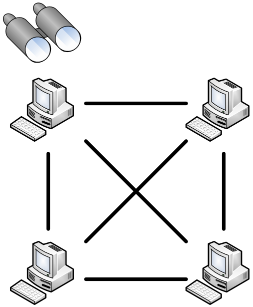
CSP Observer

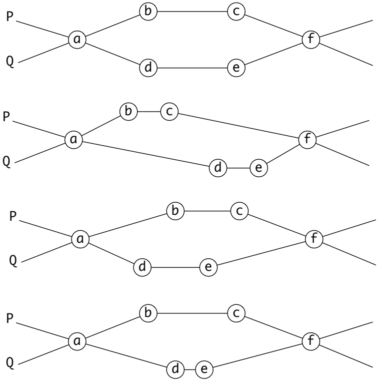


CSP Observer



CSP Observer

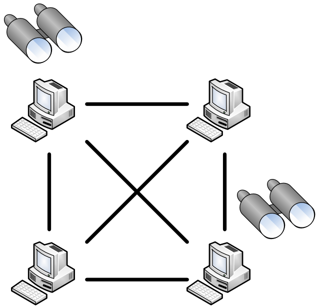




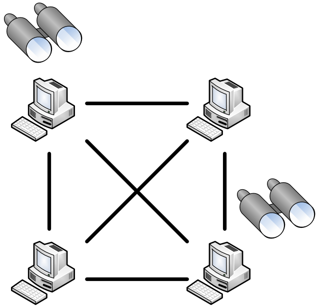
$\langle a, b, c, d, e, f \rangle$ $\langle a, b, d, c, e, f \rangle$ $\langle a, b, d, e, c, f \rangle$

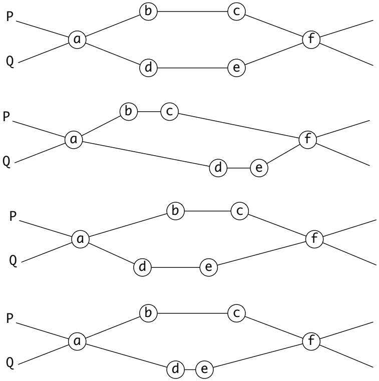
$\langle a, d, b, c, e, f \rangle$ $\langle a, d, e, b, c, f \rangle$ $\langle a, d, b, e, c, f \rangle$

VCR Observer(s)



VCR Observer(s)

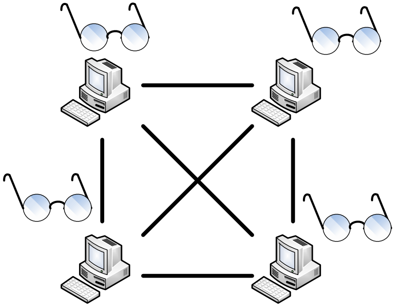




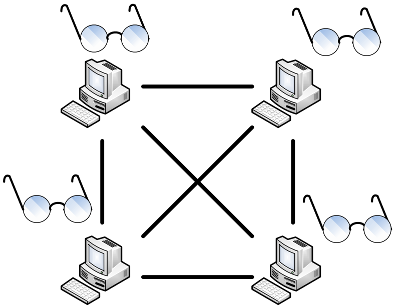
$$\langle \{a\}, \{b, d\}, \{c, e\}, \{f\} \rangle \qquad \langle \{a\}, \{b\}, \{c, d\}, \{e\}, \{f\} \rangle$$

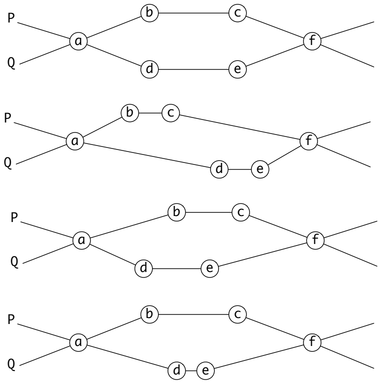
$$\langle \{a\}, \{d\}, \{b, e\}, \{c\}, \{f\} \rangle$$

Structural Observers



Structural Observers





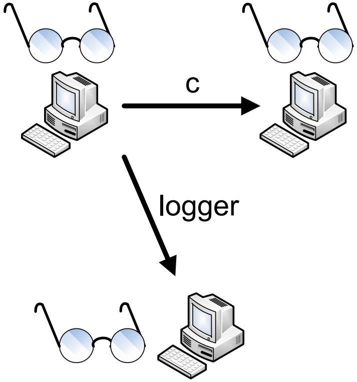
$$(a \rightarrow b \rightarrow c \rightarrow f) \parallel (a \rightarrow d \rightarrow e \rightarrow f)$$

Traces

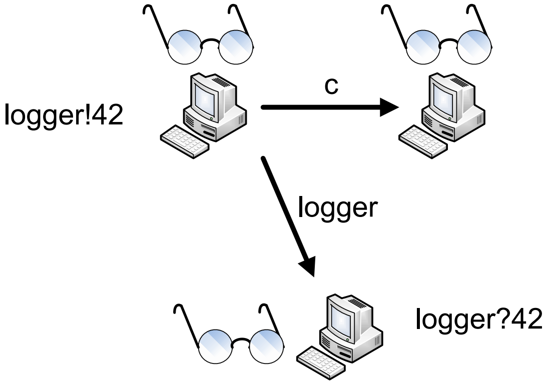
Three types of traces:

1. CSP traces
   - A sequence of individual events, recorded by the observer; events observed simultaneously are interleaved
   - abstracts away time and space
2. VCR Traces
   - A sequence of event multisets; multiple observers account for different views
   - preserves time independence; abstracts away space
3. Structural Traces
   - Sequential and parallel composition of the trace reflects the program's structure
   - preserves time independence and space

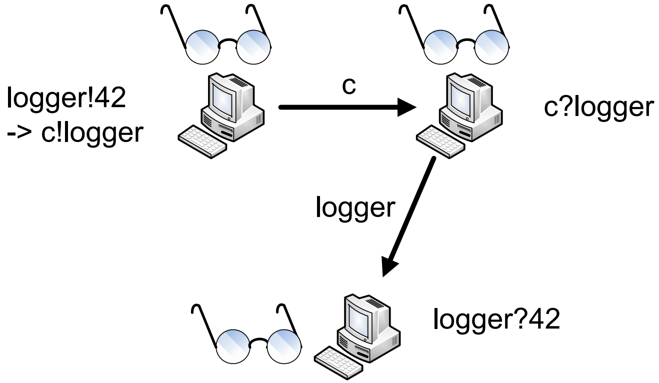
Space and Mobility



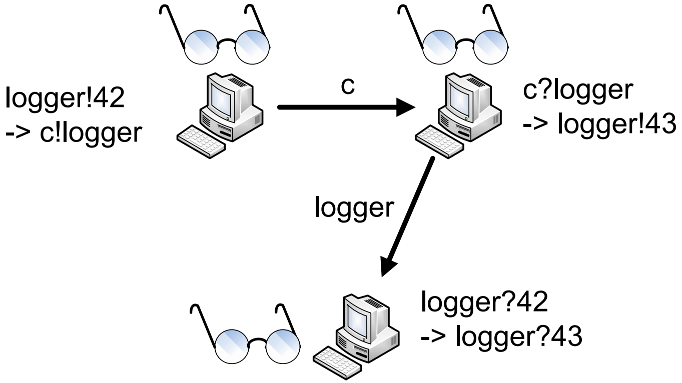
Space and Mobility



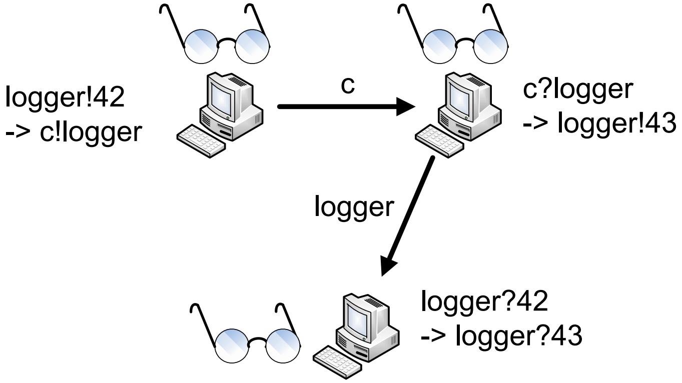
Space and Mobility



Space and Mobility



Space and Mobility



$$(\mathit{logger}!42 \rightarrow \mathit{c}!\mathit{logger}) \mid\mid (\mathit{c}?\mathit{logger} \rightarrow \mathit{logger}!43)$$
$$\mid\mid (\mathit{logger}?42 \rightarrow \mathit{logger}?43)$$

Conversion Example

CSP:

$$P = (AB \mathbin{;} AB) \underset{\{b\}}{||} (b \rightarrow b \rightarrow \text{SKIP})$$

$$\text{where } AB = (a \rightarrow \text{SKIP}) \mathbin{|||} (b \rightarrow \text{SKIP})$$

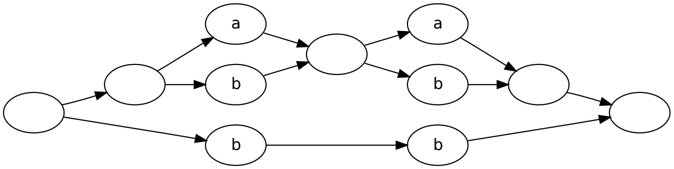
Conversion Example

CSP:

$$P = (AB \mathbin{;} AB) \underset{\{b\}}{||} (b \rightarrow b \rightarrow \text{SKIP})$$

$$\text{where } AB = (a \rightarrow \text{SKIP}) \mathbin{|||} (b \rightarrow \text{SKIP})$$

Structural Trace Visualisation:



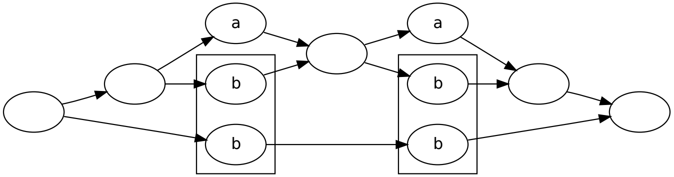
Conversion Example

CSP:

$$P = (AB \mathbin{;} AB) \underset{\{b\}}{||} (b \rightarrow b \rightarrow \text{SKIP})$$

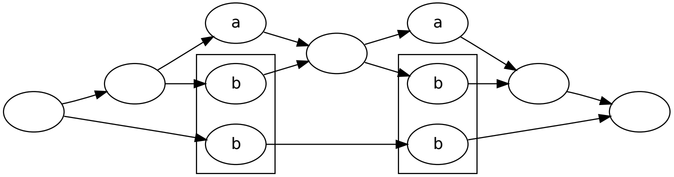
$$\text{where } AB = (a \rightarrow \text{SKIP}) \; ||| \; (b \rightarrow \text{SKIP})$$

Structural Trace Conversion Algorithm:

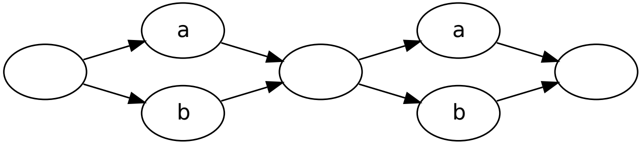


Conversion Example

Structural Trace Conversion Algorithm:

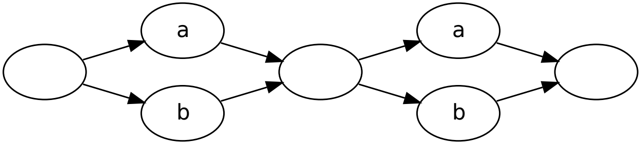


VCR Trace Visualisation:



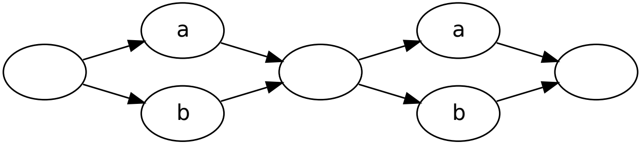
Conversion Example

VCR Trace Visualisation:

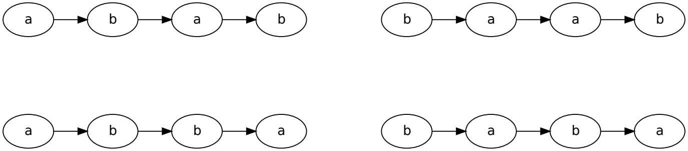


Conversion Example

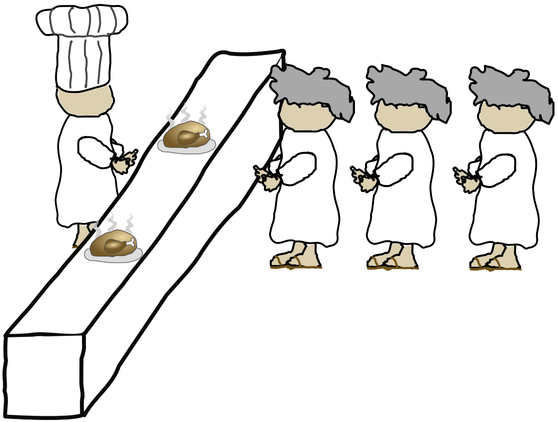
VCR Trace Visualisation:



CSP Trace Visualisation:



Starving CHP Philosophers

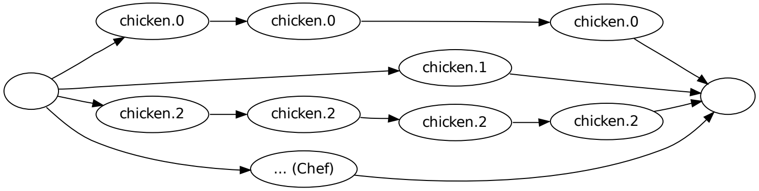


Starving CHP Philosophers Trace

```
philosopher eatChicken = forever (syncBarrier eatChicken)

chef a b c = forever  ((syncBarrier a <&> syncBarrier b)
                      <-> (syncBarrier b <&> syncBarrier c)
                      <-> (syncBarrier a <&> syncBarrier c))
```

Starving CHP Philosophers Trace

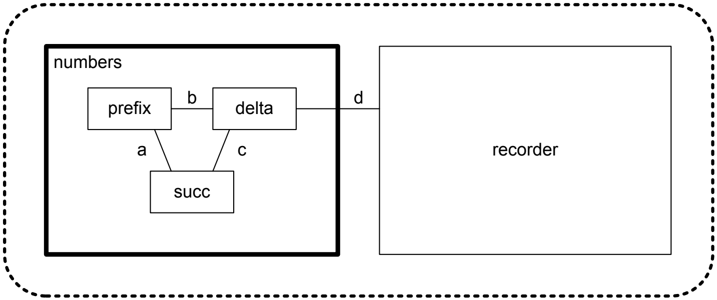
```
philosopher eatChicken = forever (syncBarrier eatChicken)

chef a b c = forever  ((syncBarrier a <&> syncBarrier b)
                       <-> (syncBarrier b <&> syncBarrier c)
                       <-> (syncBarrier a <&> syncBarrier c))
```



Summary

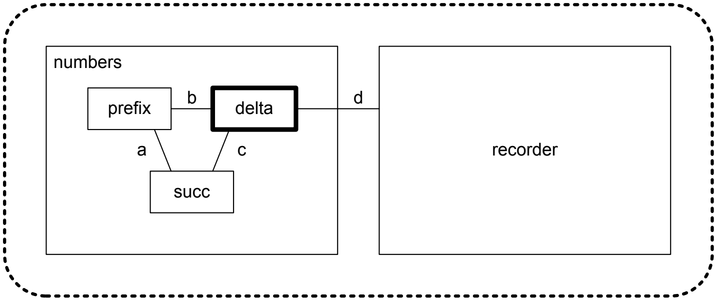
- Traces are useful for diagnostics and observing run-time behaviour
- Structural traces
  - Most straightforward and efficient to record
  - Useful for observing mobility
- Conversion algorithms
  - One Structural trace converts to many VCR traces
  - One VCR trace converts to many CSP traces
- Visualisation
  - Graphs to represent CSP, VCR and Structural traces
  - Tool support will be beneficial

Tools to Explore Traces



42*(b? -> (c! || d!))
||
(42*(b! -> a?) -> b!)
||
42*(c! -> a?)

Tools to Explore Traces



42*(b? -> (c! || d!))

Challenge: Structural traces and UTP

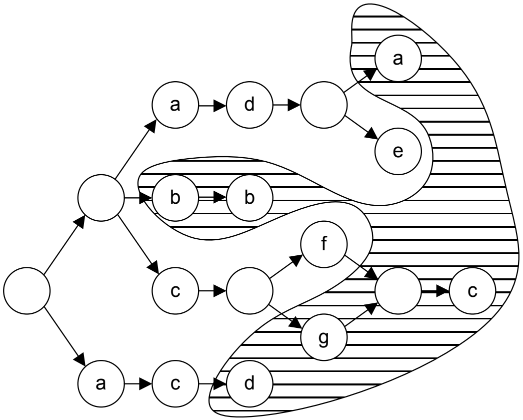


Figure: Concatenation, quotient and healthiness conditions

Questions?

- Practical demo of traces for testing – tonight at the fringe